Mitigating Tool Overuse for LLMs via Active Knowledge Boundary Probing

Zhaoyu Yang[1], **Wenjun Ke**[1,2*], **Yuanyao Li**[3], **Yuchen Liu**[3], **Junqi Xu**[5], **Peng Wang**[1,2], **Hengyuan Xu**[4]
[1]School of Computer Science and Engineering, Southeast University, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]Huawei Technologies, China
[4]College of Software Engineering, Southeast University, China
[5]Soochow University, China
{zhaoyuyang, kewenjun, pwang, xuhengyuan}@seu.edu.cn, leeyuanyao@163.com, liuyuchen46@huawei.com, 2262404058@stu.suda.edu.cn

Abstract

Tool-augmented methods aim to enhance the reasoning capabilities of large language models (LLMs) by invoking external tools, which can be broadly categorized into training-free and training-based methods. Training-free methods can directly instruct LLMs to invoke external tools, but they exhibit limited generalization in dynamic environments. In contrast, training-based methods use a teacher model to generate tool-use trajectories and fine-tune a student model to distill the trajectories that can generalize across tasks. However, they still face two major challenges: (1) Knowledge boundary mismatch: teacher trajectories may contain steps that the student cannot execute without tools due to their mismatch of knowledge boundaries. (2) Tool overuse propagation: the student inherits the misaligned tool-use strategy from teacher model. To address these challenges, we propose RADAR, an automated framework for knowledge boundary discovery and tool overuse mitigation. First, RADAR performs semantic anchoring to select representative seeds. Second, simulated annealing probing is employed to actively explore the student model's tool-use boundary and acquire sparse, verified student-aware labels. Third, it globally propagates these labels and rewrites conflicts, yielding deployment-aligned decisions with fewer unnecessary tool calls. We conduct experiments on multiple benchmarks, compared to the SOTA method, RADAR reduces average tool-use by 52.7% while improving accuracy by 3.2% across three models.

1 Introduction

Tool-augmented paradigm [Hao *et al.*, 2023; Shi *et al.*, 2025; Lu *et al.*, 2025] enhance the reasoning capabilities of large language models (LLMs) by extending their knowledge boundary across diverse tasks, such as mathematical problem reasoning [Didolkar *et al.*, 2024; Luo *et al.*, 2025], time sensitive question answering [Wang *et al.*, 2025], and human intent recognition [Harnad, 2024]. However, current LLMs have a tendency toward tool overuse. For example, Llama-3.1-8B and Mistral-7B-v0.3 invoke tools in 66.2% and 30.2% on the GSM8K benchmark, respectively, even though most can be solved without external tools [Qian *et al.*, 2025b].

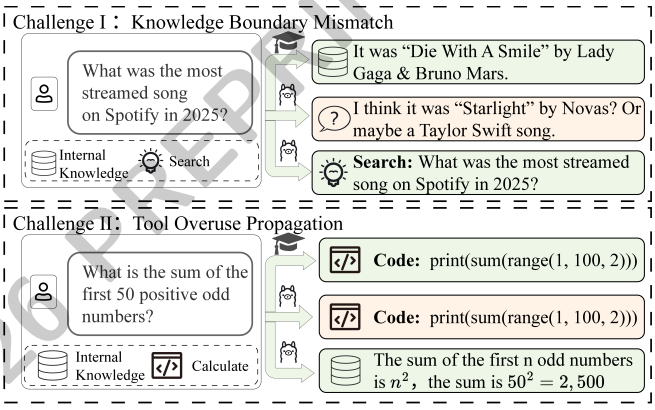

Figure 1: Comparison of teacher and student behaviors under two misalignment scenarios. (a) **Knowledge Boundary Mismatch**: missing student knowledge induces errors without tools. (b) **Tool Overuse Propagation**: redundant tool calls are distilled to the student for internally solvable tasks.

To mitigate these issues, recent studies have attempted strategies to regulate tool invocation behaviors in LLMs, which can be broadly categorized into training-free and training-based methods [Mialon *et al.*, 2023]. Training-free methods typically rely on rule-based prompting and in-context learning (ICL) constraints to invoke external tools without parameter updates [Fang *et al.*, 2025; Zhang *et al.*, 2023]. However, these methods are constrained by context window limits and sensitivity to example formatting. Moreover, they exhibit poor generalization when facing dynamic environments or unseen tool-use scenarios. For instance,

*Corresponding author.

LLMs relying on few-shot examples struggle to handle a novel problem, as they tend to apply the referenced patterns to different functional scenarios.

Training-based methods have been proposed to alleviate these challenges by fine-tuning models on tool-use related steps [Schick *et al.*, 2023; Patil *et al.*, 2024; Kong *et al.*, 2024]. Current training-based methods typically follow a knowledge distillation framework, where a teacher generates reasoning trajectories that are used to supervise a smaller student model [Ye *et al.*, 2025; Zeng *et al.*, 2025]. However, these methods still have two limitations regarding the misalignment of knowledge boundaries. On the one hand, the process overlooks the difference in parametric knowledge between the teacher and the student model. As illustrated in Figure 1(a), the teacher correctly answers *Die With A Smile* for *the most streamed song on Spotify in 2025*, but the student imitates this response and hallucinates *Starlight* due to missing knowledge. On the other hand, the student model propagates the teacher's tool overuse trajectories, consequently acquiring a bias toward invoking tools even when internal parametric knowledge would suffice [Yu *et al.*, 2023]. As shown in Figure 1(b), for *calculating the sum of the first 50 positive odd numbers*, the student propagates the teacher's tool overuse behavior, leading to unnecessary external tool calls even when the problem is solvable via internal knowledge.

In this paper, we argue that effective tool-augmented inference requires a dynamic balance between a model's internal knowledge and external tools. To achieve this balance, we first identify the student's internal knowledge boundary and then invoke external tools only when necessary to compensate for specific limitations. In light of this consideration, we propose Reasoning-boundary Anchoring & Dynamic Active Recalibration (RADAR), a three-stage framework that reconstructs the tool-use boundaries specific to the target LLMs. First, we perform semantic anchoring to build a structural foundation. Taking raw teacher-generated reasoning steps as input, we encode them into a dense embedding space and execute k-Means clustering to identify centroids, ensuring subsequent exploration starts from semantically diverse regions. Second, to address the knowledge boundary mismatch, simulated-annealing–guided active probing is performed. Using the initial anchors as the input, the student model is treated as a black-box uncertainty estimator to iteratively explore uncertain regions around the initial anchors, while newly discovered steps are labeled by a stronger model and added as new anchors to locate the decision boundary. Third, we execute global recalibration to mitigate tool overuse propagation. Taking the sparse anchors as input, we apply k-Nearest Neighbors (k-NN) propagation to extrapolate student aware tool-use strategies across the entire dataset. To ensure these tool decisions remain logically consistent within each specific query's context, we then employ the teacher model to rewrite conflicting steps. Then we utilize this data to fine-tune various models, supporting inference on downstream tasks.

We conduct experiments on the SMARTER benchmark across three domains: Math (MATH) [Hendrycks *et al.*, 2021], Time (FreshQA) [Vu *et al.*, 2024], and Intention (IN3) [Qian *et al.*, 2025a]. Extensive experiments on three datasets demonstrate that our method achieves a maximum task performance improvement of 7.4% and a maximum tool usage reduction of 75% across different models.

2 Methods

Given a dataset $\mathcal{D} = \{(q_i, \tau_i)\}_{i=1}^{N}$, each query q_i is associated with a teacher-generated intermediate reasoning sequence $\tau_i = \langle s_{i,1}, \ldots, s_{i,L_i} \rangle$ produced by the teacher model $\mathcal{M}_T$, where $s_{i,t}$ denotes the t-th step ($t \in \{1, \ldots, L_i\}$). Each step $s_{i,t}$ is annotated with a teacher tool-use label $y_{i,t}^{\mathrm{T}} = \mathbb{I}[\texttt{Tool}(s_{i,t})] \in \{0, 1\}$. Given a student model $\mathcal{M}_S$, the task is to infer a student tool decision $y_{i,t}^{\mathrm{S}} \in \{0, 1\}$ for each step, where $y_{i,t}^{\mathrm{S}} = 1$ only if the student model cannot reliably solve step $s_{i,t}$ without external tools. Formally, the output is a label set $\mathcal{Y}^{\mathrm{S}} = \{y_{i,t}^{\mathrm{S}} \mid i \in [1, N],\ t \in [1, L_i]\}$.

Figure 2 illustrates the overall architecture of RADAR. In the first stage, we select representative anchors by embedding teacher-produced reasoning steps into a latent space. In the second stage, the selected anchors are used to probe the student model to estimate the decision boundary for tool-use and assess tool necessity near the boundary. In the final stage, we propagate sparse boundary annotations via global recalibration and rewrite conflicting steps using a stronger model, producing a set of tool-use labels that is suitable for deployment with the student model.

2.1 Tool Boundary Anchoring

To ensure that the subsequent boundary probing covers the semantic space of intermediate reasoning steps, we construct a set of initial anchors. Using intermediate steps extracted from teacher trajectories as input, this stage outputs a globally distributed anchor set to initialize the probing loop and prevent sampling bias toward dense regions.

Semantic Clustering Each step $s_{i,t} \in \mathcal{S}$ is mapped into a continuous latent space by the encoder $\phi(\cdot)$, yielding an embedding vector $\mathbf{z}_{i,t} = \phi(s_{i,t}) \in \mathbb{R}^d$, where d is the embedding dimension. As illustrated in Stage#1 of Figure 2, these steps represent specific reasoning actions such as *Isolate the radical term*. Based on these, the step set $\mathcal{S}$ is partitioned into K clusters $\mathcal{C} = \{C_k\}_{k=1}^{K}$ using K-Means clustering, where each cluster satisfies $C_k \subseteq \mathcal{S}$ and has a centroid $\mathbf{c}_k \in \mathbb{R}^d$.

Initialization Seeding From each cluster C_k, a representative step is selected as an anchor by choosing the step whose embedding is closest to the corresponding centroid:

$$\hat{s}_k = \arg\min_{s \in C_k} \|\phi(s) - \mathbf{c}_k\|_2 , \tag{1}$$

The resulting anchor set is defined as $\mathcal{A}_0 = \{\hat{s}_k\}_{k=1}^{K}$. These anchors serve solely as the starting states for the boundary probing procedure in the next stage, ensuring that probing begins from semantically representative locations rather than from randomly sampled or densely concentrated regions.

2.2 Tool Boundary Probing

To identify the tool-use decision boundary of the student model within the semantic space, we conduct an active probing stage initialized from the anchor set. Starting from these seeds, we use a simulated annealing method to explore

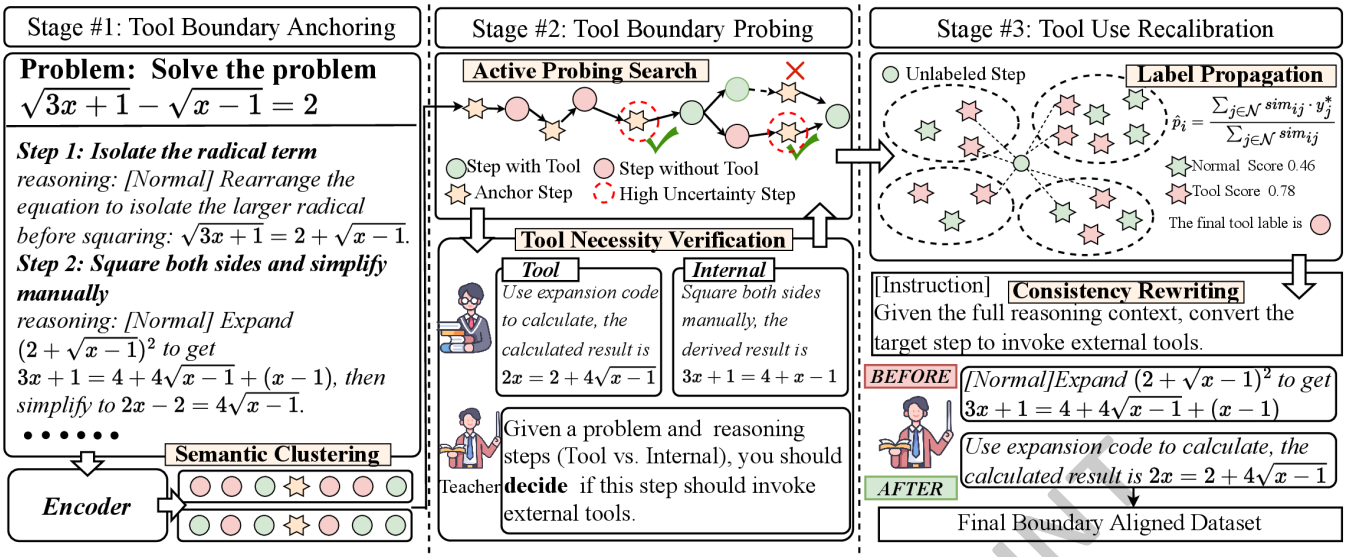

Figure 2: Overview of our RADAR workflow. The arrow indicates the data flow from previous stage to next one.

the space and iteratively select informative steps for tool-necessity queries, yielding a sparse set of boundary annotations. The procedure is summarized in Algorithm 1.

Uncertainty Quantification Unlike standard uncertainty sampling that relies on model probability outputs, we quantify ambiguity based on the local smoothness of tool-use labels in the semantic embedding space. For a candidate step s, we inspect its local neighborhood $\mathcal{N}_k(s)$ consisting of the k nearest labeled neighbors. We define the spatial entropy $\mathcal{H}(s)$ as:

$$\mathcal{H}(s) = -\sum_{y\in\{0,1\}} p(y \mid \mathcal{N}_k(s)) \log p(y \mid \mathcal{N}_k(s)), \quad (2)$$

where $p(y \mid \mathcal{N}_k(s))$ represents the proportion of neighbors with label y (tool-use or internal-reasoning). As shown in Stage#2 of Figure 2, our search prioritizes candidates where local neighbors exhibit conflicting labels, while the simulated annealing mechanism preserves a stochastic characteristic to explore other regions. A high entropy value indicates that s lies in a contentious region where tool necessity is inconsistent, signaling a decision boundary. The resulting score $\mathcal{H}(s)$ is then used as the acquisition signal in the next step to guide the simulated annealing search toward boundary-relevant candidates for tool-necessity queries.

Active Probing Search To navigate toward high uncertainty regions without becoming trapped in local optima, we model boundary probing as a stochastic search process initialized from an anchor $\hat{s} \in \mathcal{A}_0$. Let $s^{(r)}$ denote the current search state at iteration r, and let $\mathbf{z}(s) = \phi(s)$ be the embedding of step s. The next candidate $s^{(r+1)}$ is selected by maximizing an acquisition function $\mathcal{S}(\cdot)$ that balances boundary exploitation and spatial exploration:

$$\begin{aligned}\mathcal{S}(s') = \mathcal{H}(s') + \lambda_1 \cdot \mathrm{dist}(\mathbf{z}(s^{(r)}), \mathbf{z}(s')) \\ + \lambda_2 \cdot \mathbb{I}\Big(y^{\mathrm{S}}(s') \neq y^{\mathrm{S}}(s^{(r)})\Big),\end{aligned} \quad (3)$$

where λ_1 and λ_2 are weighting coefficients, $\mathrm{dist}(\cdot)$ denotes the cosine distance, and the last term rewards label flips that suggest boundary crossings. The search follows a simulated annealing strategy, in which candidates are primarily chosen to maximize $\mathcal{S}(\cdot)$, while with probability ϵ the state is restarted by sampling a step from a broad proposal distribution to improve global coverage.

Tool Necessity Verification Once a candidate step $s^{(r+1)}$ is selected, we query its student-aware tool label by a counterfactual verification procedure. Specifically, the student model $\mathcal{M}_S$ is prompted to produce two trajectories for the same step $s^{(r+1)}$: a tool-enabled trajectory $\tau^{\mathrm{tool}}(s^{(r+1)})$ and a tool-disabled trajectory $\tau^{\mathrm{void}}(s^{(r+1)})$. The teacher model $\mathcal{M}_T$ then serves as a judge $J(\cdot) \in \{0, 1\}$ to determine whether external tools are necessary:

$$y^{\mathrm{S}}(s^{(r+1)}) = J\Big(\tau^{\mathrm{tool}}(s^{(r+1)}),\, \tau^{\mathrm{void}}(s^{(r+1)})\Big), \quad (4)$$

The label is set to 1 only when $\tau^{\mathrm{tool}}(s^{(r+1)})$ provides verifiable corrections or required computations that are absent under $\tau^{\mathrm{void}}(s^{(r+1)})$, ensuring that supervision is aligned with the student model's actual limitations. For the radical equation in Stage#2 of Figure 2, the teacher identifies that the student's internal result $3x+1=4+x-1$ is wrong, whereas the tool-use version produces the correct result $2x = 2+4\sqrt{x-1}$, thereby confirming tool necessity.

2.3 Tool Use Recalibration

In the final phase, we generalize the precise but sparse boundary signals acquired from the active probing phase to the entire dataset. Through anchors identified during probing, we employ a semi supervised framework that combines label propagation with consistency aware step rewriting to extrapolate the learned boundary information to the full dataset.

Label Propagation We generalize the sparse verified labels obtained in the probing stage to the remaining unlabeled

Algorithm 1 RADAR Active Boundary Probing

Input: Anchors $\mathcal{A}_0 \subseteq \mathcal{S}$; step set $\mathcal{S}$; budget B; k; λ_1, λ_2; restart prob. ϵ.
Output: Verified set $\mathcal{L}$ with pairs $(s, y^{\mathrm{S}}(s))$.

$\mathcal{L} \leftarrow \emptyset$, $\mathcal{U} \leftarrow \mathcal{S}$, $s_{\mathrm{curr}} \sim \mathcal{A}_0$ ▷ $\mathcal{U}$: unlabeled pool
while $|\mathcal{L}| < B$ **do**
  $\mathcal{C} \leftarrow \mathrm{Cand}(s_{\mathrm{curr}}, \mathcal{U})$ ▷ local candidates near s_{curr}
  for each $s' \in \mathcal{C}$ **do**
    $\mathcal{H}(s') \leftarrow$ Eq. (3) using $\mathrm{kNN}(s', \mathcal{L}, k)$
    $\mathcal{S}(s') \leftarrow$ Eq. (4) ▷ flip via pseudo labels
  end for
  $s^* \leftarrow \arg\max_{s' \in \mathcal{C}} \mathcal{S}(s')$ ▷ most informative step
  $\tau^{\mathrm{tool}} \leftarrow \mathcal{M}_S(\mathrm{ForceTool}(s^*))$
  $\tau^{\mathrm{void}} \leftarrow \mathcal{M}_S(\mathrm{ForceInternal}(s^*))$
  $y^{\mathrm{S}}(s^*) \leftarrow J(\tau^{\mathrm{tool}}, \tau^{\mathrm{void}})$ ▷ 1 iff tools are necessary
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(s^*, y^{\mathrm{S}}(s^*))\}$; $\mathcal{U} \leftarrow \mathcal{U} \setminus \{s^*\}$ ▷ update
  if $\mathrm{Rand}() < \epsilon$ **then**
    $s_{\mathrm{curr}} \sim \mathcal{U}$ ▷ restart for coverage
  else
    $s_{\mathrm{curr}} \leftarrow s^*$ ▷ local move
  end if
end while
return $\mathcal{L}$

steps. Let $\mathcal{L}$ denote the current labeled step set storing pairs $(s, y^{\mathrm{S}}(s))$, and let $\mathcal{U} = \mathcal{S} \setminus \{s \mid (s, y^{\mathrm{S}}(s)) \in \mathcal{L}\}$ be the unlabeled step set. For each $s \in \mathcal{U}$, we estimate the posterior probability of requiring external tools via a similarity-weighted k-Nearest Neighbors (k-NN) rule:

$$\hat{p}\big(y^{\mathrm{S}}(s) = 1 \mid s\big) = \frac{\sum_{s' \in \mathcal{N}_k(s)} \mathrm{sim}(\mathbf{z}(s), \mathbf{z}(s')) \cdot y^{\mathrm{S}}(s')}{\sum_{s' \in \mathcal{N}_k(s)} \mathrm{sim}(\mathbf{z}(s), \mathbf{z}(s'))}, \tag{5}$$

where $\mathbf{z}(s) = \phi(s)$ is the step embedding and $\mathrm{sim}(\cdot, \cdot)$ denotes cosine similarity. This weighting emphasizes semantically closer neighbors and prioritizes precision when predicting tool necessity. To mitigate tool overuse, we adopt internal reasoning for steps with insufficient neighbor support.

Consistency Rewriting The recalibrated tool-use labels $\mathcal{Y}^{\mathrm{S}}$ may be inconsistent with the original intermediate reasoning text in τ_i. Since the reasoning content must be updated to align with these new tool-use decisions, we perform a filling process to regenerate the relevant steps. To ensure that the modified steps remain logically coherent with the original problem and the surrounding reasoning context, we adopt a consistency rewriting method where the teacher model serves as an editor. As illustrated in Stage #3 of Figure 2, the teacher transforms manual calculations into precise tool-invocation steps, maintaining logical coherence while matching the student model's specific capability boundary. This rewriting produces a label-consistent dataset $\mathcal{D}^*$ derived from $(\mathcal{D}, \mathcal{Y}^{\mathrm{S}})$ for downstream fine-tuning, while the task output remains the student reasoning steps set $\mathcal{Y}^{\mathrm{S}}$.

3 Experiments

3.1 Experimental Settings

Datasets and Metrics We evaluate our RADAR on three SMART-ER domains: Math (MATH) [Hendrycks *et al.*, 2021], Time (FreshQA) [Vu *et al.*, 2024], and Intention (IN3) [Qian *et al.*, 2025a], which assess step-level reasoning between parametric reasoning and tool-use. For Math and Time, we report accuracy and average tool calls per query. For Intention, we use the original metrics: summarized intention coverage and tool calls per query. Together, these metrics assess both answer quality and tool-use appropriateness.

Implementation and Tools We select Mistral-7B-v0.3, Meta-Llama-3.1-8B, and Mistral-Nemo-12B-2407 as student models, with GPT-4o serving as the teacher, and BAAI/bge-large-en-v1.5 for embeddings. We provide three tools: **Code** (a Python interpreter), **Search** (the Serper API), and **Ask User** (simulated by GPT-4o for the Intention domain).

Baselines We compare RADAR with state-of-the-art training-free and training-based baselines. (1) *Training-free baselines*: zero-shot [Kojima *et al.*, 2022] and tool-use [Gao *et al.*, 2023]. (2) *Training-based baselines*: CoT fine-tuning [Wei *et al.*, 2022] and SMART [Qian *et al.*, 2025b].

3.2 Main Results

The experimental results are presented in Table 1. Our observations lead to the following conclusions.

First, Reasoning Prompt achieves competitive performance without any tool invocation among training-free methods, yielding Math accuracy of 17.25%, 53.00%, and 47.00% and Time accuracy of 29.00%, 26.00%, and 34.00% for the three models, respectively. These results suggest that parametric reasoning alone provides a usable baseline, yet remains insufficient for domains that inherently benefit from external information. Compared to the training-free methods, training-based SMARTAgent further improves task performance over reasoning-only prompting, but at the cost of substantially more tool usage, requiring 0.60, 0.88, and 0.82 tool calls on Math, 1.00, 1.05, and 1.00 on Time, and 3.60, 3.80, and 3.34 on Intention, respectively. This indicates that supervision can enhance tool-augmented reasoning, but still tends to induce tool dependence, especially for intention-oriented queries.

Second, RADAR achieves consistently stronger results than SMARTAgent while using fewer tools across all domains. RADAR achieves Math accuracy of 30.15%, 56.15%, and 49.75% with only 0.44, 0.62, and 0.61 tool calls, reaches Time accuracy of 67.00%, 72.00%, and 70.00% with 0.89, 0.95, and 0.83 tool calls. Moreover, RADAR attains Intention coverage of 87.27%, 82.07%, and 85.11% while reducing tool usage to 1.50, 0.93, and 0.84, respectively. These results demonstrate that RADAR achieves superior performance while reducing the dependency on external tools.

Overall, the simultaneous gains in accuracy/coverage and the reduction in tool calls demonstrate that RADAR more precisely identifies when tools are truly necessary, yielding a better performance-efficiency trade-off than SOTA supervision.

Method	Model	**Math** (MATH)		**Time** (FreshQA)		**Intention** (IN3)	
		tool-use (↓)	Acc (%)	tool-use (↓)	Acc (%)	tool-use (↓)	Coverage (%)
Training-free Methods							
Reasoning Prompt [Kojima *et al.*, 2022]	Mistral-7B-Instruct-v0.3	0.00	17.25	0.00	29.00	0.00	–
	Meta-Llama-3.1-8B-Instruct	0.00	53.00	0.00	26.00	0.00	–
	Mistral-Nemo-Instruct-2407	0.00	47.00	0.00	34.00	0.00	–
Tool Prompt [Gao *et al.*, 2023]	Mistral-7B-Instruct-v0.3	3.90	13.25	1.67	49.00	3.80	63.04
	Meta-Llama-3.1-8B-Instruct	1.93	51.00	2.05	56.00	3.77	70.20
	Mistral-Nemo-Instruct-2407	2.35	46.00	1.90	53.00	1.80	59.27
Training-based Methods							
Normal Reasoning [Wei *et al.*, 2022]	Mistral-7B-Instruct-v0.3	0.00	17.00	0.00	48.00	0.00	–
	Meta-Llama-3.1-8B-Instruct	0.00	41.00	0.00	48.00	0.00	–
SMARTAgent [Qian *et al.*, 2025b]	Mistral-7B-Instruct-v0.3	<u>0.60</u>	<u>22.75</u>	<u>1.00</u>	<u>64.00</u>	<u>3.60</u>	<u>81.76</u>
	Meta-Llama-3.1-8B-Instruct	<u>0.88</u>	<u>54.75</u>	<u>1.05</u>	<u>67.00</u>	<u>3.80</u>	<u>78.28</u>
	Mistral-Nemo-Instruct-2407	<u>0.82</u>	<u>49.50</u>	<u>1.00</u>	**70.00**	<u>3.34</u>	<u>82.30</u>
RADAR (Ours)	Mistral-7B-Instruct-v0.3	**0.44** (-0.16↓)	**30.15** (+7.40↑)	**0.89** (-0.11↓)	**67.00** (+3.00↑)	**1.50** (-2.10↓)	**87.27** (+5.51↑)
	Meta-Llama-3.1-8B-Instruct	**0.62** (-0.26↓)	**56.15** (+1.40↑)	**0.95** (-0.10↓)	**72.00** (+5.00↑)	**0.93** (-2.87↓)	**82.07** (+3.79↑)
	Mistral-Nemo-Instruct-2407	**0.61** (-0.21↓)	**49.75** (+0.25↑)	**0.83** (-0.17↓)	**70.00** (+0.00↑)	**0.84** (-2.50↓)	**85.11** (+2.81↑)
Closed-Source Models							
Reasoning Prompt [Kojima *et al.*, 2022]	GPT-4o-mini	0.00	**73.00**	0.00	44.00	0.00	–
	GPT-4o	0.00	**79.50**	0.00	47.00	0.00	–
Tool Prompt [Gao *et al.*, 2023]	GPT-4o-mini	2.55	54.50	1.06	**56.00**	1.91	76.44
	GPT-4o	0.27	79.25	1.01	**65.00**	1.17	86.80

Table 1: Main results on the SMART-ER benchmark. **Bold** and <u>underlined</u> values denote the best and second-best performance among functional models, respectively. Green arrows (↑/↓) indicate improvements relative to the best-performing baseline.

3.3 Ablation Experiments

To explore the contribution of each component in the proposed method, we conduct ablation experiments on the two datasets and display the results in Table 2. To quantify the fundamental trade-off between supervision quality and boundary stability, we introduce the Quality-Stability Index (QSI), defined as the ratio of Supervision F1 to the Flip Ratio (F1/Flip). A higher QSI indicates a more robust alignment that maintains high correctness with minimal decision oscillation. The performance decline follows the order: RADAR w/o AM $>$ RADAR w/o SP $>$ RADAR w/o SC.

About semantic clustering (SC) Relative to other variants, RADAR w/o SC exhibits inferior stability, with a higher flip ratio across datasets. On the Math domain, the flip ratio reaches 19.8%, exceeding the 18.1% of the full framework. This variant reduces the Efficiency Cost to 26,940 by bypassing clustering but decreases the Overall QSI to 3.72. Removing SC weakens global coverage and lowers the SNR, yielding less stable boundary estimation.

About soft propagation (SP) Relative to other variants, RADAR w/o SP exhibits pronounced instability, as reflected by a substantially higher flip ratio. On the Time domain, the flip ratio increases to 30.2. Hard voting triggers overconfident label propagation. This variant raises the Efficiency Cost to 42,210 on Math, proving that soft propagation maintains Quality while ensuring cost-effectiveness.

About annealing method (AM) Relative to other variants, RADAR w/o AM exhibits the poorest overall performance. On the Math domain, the supervision Quality measured by F1 drops to 2.8. Removing the annealing mechanism traps the probing process in suboptimal regions, reducing the SNR to 0.08 and increasing Supervision Noise to 34.7. These factors lead to a negligible Overall QSI of 0.21.

3.4 Analysis Experiments

Label Efficiency and Annotation Quality We evaluate the quality of the generated labels against a student-derived oracle. Figure 3a shows that teacher distillation achieves a label accuracy of only 65.1% on Math. In contrast, RADAR significantly improves label accuracy to 89.7%, 95.5%, and 93.0% on Math, Time, and Intention, respectively. Figure 3b further demonstrates high annotation efficiency: label accuracy on the Time domain exceeds 91% with only a 7.5% anchor budget. Moreover, RADAR yields a 1.63× speedup, reducing the average annotation cost to 61.3

Variant	Efficiency	Stability		Quality		Overall
	Cost (↓)	Flip (↓)	SNR (↑)	Sup. F1 (↑)	Noise (↓)	QSI (↑)
Dataset: Math						
w/o SC	**26,940**	19.8	2.70	73.8	27.3	3.72
w/o AM	23,460	13.5	0.08	2.8	34.7	0.21
w/o SP	42,210	24.2	**3.35**	**80.7**	**24.1**	3.33
RADAR	31,530	**18.1** (-1.7↓)	2.41	71.2	29.6	**3.93** (+0.21↑)
Dataset: Time						
w/o SC	11,448	22.5	**4.02**	**81.7**	**20.3**	3.63
w/o AM	11,289	20.0	0.14	7.0	51.5	0.35
w/o SP	11,946	30.2	2.85	79.1	27.8	2.61
RADAR	**11,383** (-65↓)	**21.5** (-1.0↓)	3.36	79.2	23.6	**3.68** (+0.05↑)

Table 2: Ablation study results. **Bold** and underlined values denote the best and second-best performance among functional variants. Green arrows indicate RADAR's improvement over the strongest valid baseline. Note that *w/o AM* is excluded from rankings due to model collapse (Supervision F1 ≤ 7.0).

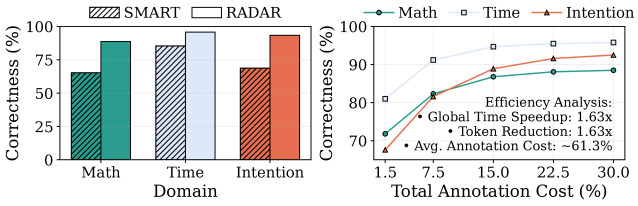

Figure 3: Supervision quality and efficiency analysis. (a) Comparison of supervision correctness. (b) Supervision correctness and cost efficiency across different budgets.

Semantic Stability and Local Continuity To assess local semantic continuity, we apply domain-specific perturbation operators to step names in the Math and Time domains and measure the resulting embedding drift. Figure 4 reports the Cumulative Distribution Function (CDF) of cosine distances between original and perturbed embeddings. Single operators cause minor shifts, while stacked operators yield a monotonic increase in distance, indicating that the embedding space preserves fine-grained structure and maps incremental text changes to predictable semantic drift.

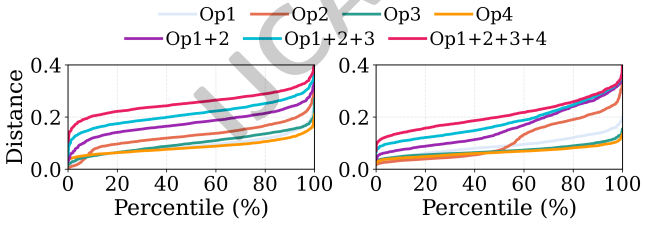

Figure 4: The plots display the CDF of cosine distances between original and perturbed step names. For **Math**, Op1 represents Operand, Op2 Operator, Op3 Verb, and Op4 Mode. For **Time**, Op1 represents Entity, Op2 Action, Op3 Temporal, and Op4 Mode.

Validity of Entropy as a Proxy To assess our active selection premise, we validate neighbor entropy as a proxy for semantic uncertainty by examining its alignment with decision

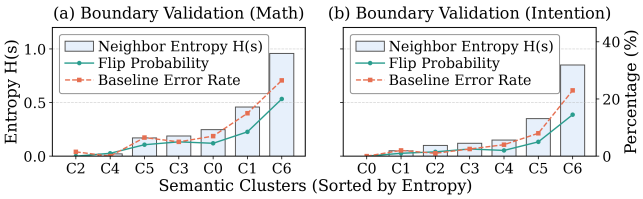

Figure 5: Correlation between neighbor entropy and decision uncertainty. Higher entropy levels (bars) align with increased label flip probabilities (solid lines) and baseline error rates (dashed lines) across both (a) Math and (b) Intention domains.

instability and task difficulty. We expect a useful proxy to correlate with label flip probability [Settles, 2009] and baseline error rates. As shown in Figure 5, entropy-stratified clusters exhibit a clear trend: higher entropy consistently corresponds to higher flip probability and higher error. For example, in Math, high-entropy clusters show label flip probability above 15% and baseline error around 25%, and a similar pattern is observed in Intention, where the highest-entropy cluster coincides with the peak error rate. These results support neighbor entropy as an effective difficulty indicator for targeting informative regions without global supervision.

Sensitivity Analysis of Initial Cluster Count We assess robustness to initialization by varying the number of semantic clusters used for anchor seeding from 10 to 50. As shown in Figure 6, the correctness of tool-use label supervision on Math, Time, and Intention remains nearly unchanged, with variations below 1%. Task performance is similarly stable across this range. For example, Intention accuracy stays between 82% and 84%. This indicates the selected anchors reliably capture the global boundary structure.

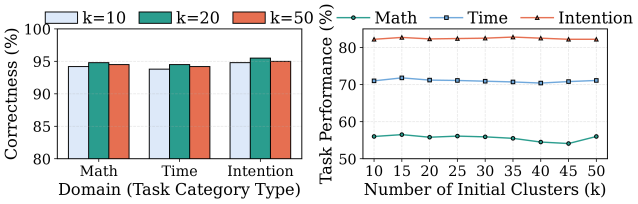

Figure 6: Robustness analysis of initialization granularity. Varying the number of initial clusters (k) has negligible impact on (a) supervision correctness and (b) fine-tuning task performance, confirming the stability of the framework.

Visualization of Decision Boundaries We visualize student-specific boundaries by projecting step embeddings with UMAP [McInnes *et al.*, 2018] and overlaying tool-use densities for Meta-Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3. As shown in Figure 7, both models rely on tools for complex geometry tasks in cluster4, with tool-use rates of 28.7% and 37.2%, but diverge sharply on basic area calculations in cluster3, where Mistral uses tools for 44.2% of steps compared to only 1.1% for Llama. This confirms that model-specific knowledge boundaries exist and serve as the driver for different tool-invocation behaviors.

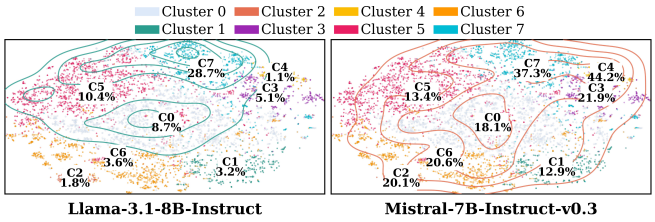

Figure 7: Visualization of model-specific decision boundaries. The distinct tool-use density patterns confirm that RADAR captures unique capability fingerprints within the same semantic space.

3.5 Case Study

Impact of Boundary Alignment Table 3 demonstrates the reasoning change resulting from boundary alignment on an algebraic task. The baseline SMART fails to invoke tools for the error-prone algebraic simplification, yet redundantly invokes a calculator to verify simple domain constraints. In contrast, RADAR aligns its execution with capability boundaries by invoking external tools to accurately derive the quadratic form $x^2 - 23x + 90 = 0$. Crucially, it successfully reverts to internal reasoning to identify the extraneous root $x = 5$, proving it can handle logical consistency checks without external tools. Such switching illustrates the balance between internal reasoning and external tool invocation and ensures both arithmetic precision and logical consistency.

SMART/RADAR	Intermediate Reasoning Steps
Query: Solve $\sqrt{9x-41} = x-7$ for real solutions, ensuring consistency with the domain constraint $x \geq 7$.	
Step 1 Correct(✗, ✓) Tooluse(✗, ✓)	**[Tool]** [Code](Simply the Equation). Simplify and rearrange terms to derive the quadratic $x^2 - 23x + 90 = 0$.
Step 2 Correct(✗, ✓) Tooluse(✗, ✗)	**[Normal]** Solve the quadratic equation $x^2 - 23x + 90 = 0$ via factorization to identify the candidate set $\{5, 18\}$.
Step 3 Correct(✓, ✓) Tooluse(✓, ✗)	**[Normal]** Check the domain constraint $x \geq 7$. Discards $x = 5$ as extraneous and $x = 18$ is the valid solution.

Table 3: Step Strategy Comparison (SMART vs. RADAR). The table shows the correct reasoning path generated by RADAR. ✓/✗ indicate whether the step is correct and whether a tool was used.

Validation of Tool Necessity Table 4 provides evidence for the necessity of external tools through enforced execution. The Original column reveals that dynamic queries, such as the unemployment rate in Case 1 and box office figures in Case 3, induce hallucinations due to outdated memory, confirming the necessity of tools. In contrast, the question in Case 2 is answered correctly by internal reasoning, proving that external search is redundant. This ensures that the tool is invoked only when necessary, optimizing inference accuracy.

Case 1: *Query: Unemployment rate in Brandenburg (Jan 2025).*	
Original	**[Normal]** Based on historical trends from late 2024, the rate is likely around 5.8%.
Refined	**[Tool]** Search(Unemployment rate brandenburg Jan 2025). Official data reports the rate rose to 6.3%.
Case 2: *Query: Country with Maputo as capital (2024 upheavals).*	
Original	**[Tool]** Search(Maputo 2024 news). Discusses the recent election crisis and protests, failing to state the capital.
Refined	**[Normal]** Maputo is the capital city of Mozambique, located on the Indian Ocean coast of Southeast Africa.
Case 3: *Query: Latest box office for the Mufasa backstory movie.*	
Original	**[Normal]** Refers to the 2019 movie which grossed over $1.6B worldwide.
Refined	**[Tool]** Search(Mufasa: The Lion King 2024 box office). The 2024 prequel has grossed over $180M globally.

Table 4: Tool Necessity Validation via Evaluation of the counterfactual, where green denotes valid detection and red indicates hallucinations or outdated errors.

4 Related Work

Tool Learning for LLMs External tools augment LLMs with real-time data and specialized computation [Schick *et al.*, 2023; Qin *et al.*, 2024], pioneered by ReAct's reasoning-action integration [Yao *et al.*, 2023]. These methods form two main paradigms. Training-free approaches leverage in-context learning [Yao *et al.*, 2023; Liu *et al.*, 2024; Fang *et al.*, 2025] but suffer from high context overhead and limited generalization. Conversely, training-based methods internalize tool-use via supervised fine-tuning [Schick *et al.*, 2023; Patil *et al.*, 2024; Chen *et al.*, 2023; Tang *et al.*, 2023], with recent reinforcement learning techniques further optimizing tool-call timing to prevent overuse [Jin *et al.*, 2025; Li *et al.*, 2025b].

Knowledge Boundaries for LLMs Research on knowledge boundaries examines LLMs' ability to recognize the limits of their parametric memory [Li *et al.*, 2025a]. Boundary signals are derived from probability calibration, internal probing, and consistency checks [Kadavath *et al.*, 2022; Tian *et al.*, 2023; Manakul *et al.*, 2023]. Detecting these limits enables test-time interventions like selective abstention or adaptive retrieval [Jiang *et al.*, 2023; Asai *et al.*, 2024], reducing hallucinations and improving model reliability [Yin *et al.*, 2023; Mialon *et al.*, 2023].

5 Conclusion

In this paper, we propose RADAR, a student-aware framework for tool supervision in distilled LLMs. We reveal that unreliable tool supervision largely stems from the knowledge boundary mismatch between teacher and student models. Through a three-stage pipeline, RADAR performs semantic anchoring to select representative seeds, conducts simulated-annealing–based active probing to acquire sparse verified student-aware labels, and applies global recalibration with consistency rewriting to propagate boundary information and suppress tool overuse. Experimental results demonstrate the superiority of RADAR over existing baselines.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work was supported by the National Science Foundation of China (Grant No. 62376057), the Start-up Research Fund of Southeast University (RF1028623234), and the Big Data Computing Center of Southeast University.

References

[Asai *et al.*, 2024] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[Chen *et al.*, 2023] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

[Didolkar *et al.*, 2024] Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Jimenez Rezende, Yoshua Bengio, Michael C Mozer, and Sanjeev Arora. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:19783–19812, 2024.

[Fang *et al.*, 2025] Wei Fang, Yang Zhang, Kaizhi Qian, James R. Glass, and Yada Zhu. PLAY2PROMPT: Zero-shot tool instruction optimization for LLM agents via tool play. In *Findings of the Association for Computational Linguistics (ACL)*, pages 26274–26290, 2025.

[Gao *et al.*, 2023] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10764–10799. PMLR, 2023.

[Hao *et al.*, 2023] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:45870–45894, 2023.

[Harnad, 2024] Stevan Harnad. Language writ large: Llms, chatgpt, grounding, meaning and understanding. *arXiv preprint arXiv:2402.02243*, 2024.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:1584–1603, 2021.

[Jiang *et al.*, 2023] Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7969–7992, 2023.

[Jin *et al.*, 2025] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[Kadavath *et al.*, 2022] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[Kojima *et al.*, 2022] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:22199–22213, 2022.

[Kong *et al.*, 2024] Yilun Kong, Jingqing Ruan, YiHong Chen, Bin Zhang, Tianpeng Bao, Shi Shiwei, Du Guo Qing, Xiaoru Hu, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, and Xueqian Wang. TPTU-v2: Boosting task planning and tool usage of large language model-based agents in real-world industry systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP)*, pages 371–385, 2024.

[Li *et al.*, 2025a] Moxin Li, Yong Zhao, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, Tat-Seng Chua, and Yang Deng. Knowledge boundary of large language models: A survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5131–5157, 2025.

[Li *et al.*, 2025b] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[Liu *et al.*, 2024] Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*, 2024.

[Lu *et al.*, 2025] Hailun Lu, Xingming Li, Xuanyu Ji, Zhigang Kan, and Qingyong Hu. Toolfive: Enhancing tool-augmented llms via tool filtering and verification. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[Luo *et al.*, 2025] Haipeng Luo, Huawen Feng, Qingfeng Sun, Can Xu, Kai Zheng, Yufei Wang, Tao Yang, Han Hu, Yansong Tang, and Di Wang. Agentmath: Empowering mathematical reasoning for large language models via tool-augmented agent. *arXiv preprint arXiv:2512.20745*, 2025.

[Manakul *et al.*, 2023] Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empiri-*

cal Methods in Natural Language Processing (EMNLP), pages 9004–9017, 2023.

[McInnes *et al.*, 2018] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[Mialon *et al.*, 2023] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

[Patil *et al.*, 2024] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:126544–126565, 2024.

[Qian *et al.*, 2025a] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

[Qian *et al.*, 2025b] Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. SMART: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics (ACL)*, pages 4604–4621, 2025.

[Qin *et al.*, 2024] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[Schick *et al.*, 2023] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:68539–68551, 2023.

[Settles, 2009] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[Shi *et al.*, 2025] Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*, pages 2222–2237, 2025.

[Tang *et al.*, 2023] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

[Tian *et al.*, 2023] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5433–5442, 2023.

[Vu *et al.*, 2024] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. Freshllms: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics*, pages 13697–13720, 2024.

[Wang *et al.*, 2025] Xiaohan Wang, Dian Li, Yilin Zhao, and Gang Liu. Metatool: Facilitating large language models to master tools with meta-task augmentation. In *Proceedings of the 2nd ACM Workshop in AI-powered Question & Answering Systems*, pages 3–11, 2025.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24824–24837, 2022.

[Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[Ye *et al.*, 2025] Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Tao Ji, Qi Zhang, et al. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, pages 156–187, 2025.

[Yin *et al.*, 2023] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics (ACL)*, pages 8653–8665, 2023.

[Yu *et al.*, 2023] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):150–170, 2023.

[Zeng *et al.*, 2025] Yirong Zeng, Xiao Ding, Yuxian Wang, Weiwen Liu, Yutai Hou, Wu Ning, Xu Huang, Duyu Tang, Dandan Tu, Bing Qin, et al. itool: Reinforced fine-tuning with dynamic deficiency calibration for advanced tool use. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13901–13916, 2025.

[Zhang *et al.*, 2023] Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. Syntax error-free and generalizable tool use for llms via finite-state decoding. *arXiv preprint arXiv:2310.07075*, 2023.